2021-09

AN EVALUATION OF DE-ANONYMIZATION ATTACKS AGAINST PHYSICAL DOWNLINK SHARED CHANNEL DATA IN 5G NEW RADIO

Garrett, Jacob M.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

AN EVALUATION OF DE-ANONYMIZATION ATTACKS AGAINST PHYSICAL DOWNLINK SHARED CHANNEL DATA IN 5G NEW RADIO

by

Jacob M. Garrett

September 2021

Thesis Advisor: John D. Roth
Second Reader: John C. McEachen

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE AN EVALUATION OF DE-ANONYMIZATION ATTACKS AGAINST PHYSICAL DOWNLINK SHARED CHANNEL DATA IN 5G NEW RADIO	**5. FUNDING NUMBERS**
6. AUTHOR(S) Jacob M. Garrett	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The development and adoption of 5G technology is rapidly progressing worldwide, and users of this technology are expected to increase to three billion by 2026. Third Generation Partnership Project (3GPP) has designed the specifications of this new network infrastructure with security in mind; however, ongoing research continues to verify and scrutinize the system to help keep the data traveling this network safe. This thesis evaluates the feasibility and performance of three different attacks on the anonymity of low-density parity-check (LDPC) encoded data in 5G physical downlink shared channel (PDSCH), then compares the findings to a similar study done of Polar coded data in physical downlink control channel (PDCCH). The anonymity attacks make use of a scrambling sequence, which is associated to a unique radio network temporary identifier (RNTI) and added to encoded data before transmission. Brute-force attacks, known plaintext attacks, and analysis using multiple messages were used to explore the security of this scrambling sequence and RNTI. This paper finds the standards used in PDSCH to provide more robust security than PDCCH and recommends the adaptation of parameters used in PDSCH in future applications.

14. SUBJECT TERMS 5G, anonymity, error correction coding, low-density parity-check, LDPC, physical downlink control channel, PDCCH, radio network temporary identifier, RNTI	**15. NUMBER OF PAGES** 59
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

AN EVALUATION OF DE-ANONYMIZATION ATTACKS AGAINST PHYSICAL DOWNLINK SHARED CHANNEL DATA IN 5G NEW RADIO

Jacob M. Garrett
Lieutenant, United States Navy
BS, United States Naval Academy, 2016

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2021

Approved by: John D. Roth
Advisor

John C. McEachen
Second Reader

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The development and adoption of 5G technology is rapidly progressing worldwide, and users of this technology are expected to increase to three billion by 2026. Third Generation Partnership Project (3GPP) has designed the specifications of this new network infrastructure with security in mind; however, ongoing research continues to verify and scrutinize the system to help keep the data traveling this network safe. This thesis evaluates the feasibility and performance of three different attacks on the anonymity of low-density parity-check (LDPC) encoded data in 5G physical downlink shared channel (PDSCH), then compares the findings to a similar study done of Polar coded data in physical downlink control channel (PDCCH). The anonymity attacks make use of a scrambling sequence, which is associated to a unique radio network temporary identifier (RNTI) and added to encoded data before transmission. Brute-force attacks, known plaintext attacks, and analysis using multiple messages were used to explore the security of this scrambling sequence and RNTI. This paper finds the standards used in PDSCH to provide more robust security than PDCCH and recommends the adaptation of parameters used in PDSCH in future applications.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

3GPP	Third Generation Partnership Project
5G	5th Generation New Radio
CRC	Cyclic Redundancy Check
DCI	Downlink Control Information
GDPR	General Data Protection Regulation
LDPC	Low-Density Parity Check
NPS	Naval Postgraduate School
PDCCH	Physical Downlink Control Channel
PDSCH	Physical Downlink Shared Channel
RAM	Random Access Memory
ROM	Read-Only Memory
RNTI	Radio Network Temporary Identifier
UE	User Equipment

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 The Promise of 5G

In the last three years cellular data traffic has nearly tripled, and is projected to double again by 2026 [2]. The worldwide demand for increased mobile data speed, volume, and coverage has driven several generations of cellular technologies. The next standard for widespread cellular communications to be implemented is called 5th Generation New Radio (5G). The publishing body Third Generation Partnership Project (3GPP) has released specifications for networks that can support an enormous number of connected devices at data rates that have not been possible for large-scale cellular networks before. Changes in governance have also made large portions of the electromagnetic spectrum available in frequency ranges that were previously reserved. This change now allows for the realization of 5G standards in ways that maximize their utility.

The vision for this highly capable network is just as advanced. The network was engineered to support a large number of wearable devices on its users, autonomous vehicles, and millions of Internet of Things (IoT) devices dispersed around urban landscapes all at once. With this vision realized, the dependence of financial, infrastructural, safety, entertainment, and even governmental systems on the availability and security of this network will be unquestionable. So much data has enormous value to researchers, to its users, and unfortunately to criminals as well. The electromagnetic spectrum is accessible to anyone with a properly configured radio, and this inherently makes privacy a challenge. The focus on protecting personal privacy and data emerging in modern society can be palpably felt in legislation like the European Union General Data Protection Regulation (GDPR) [3]. This drive has led to new innovations to achieve private communications in a public medium. In order to properly adopt this new technology in safe and secure ways, each and every aspect of the proposed system must be thoroughly examined and this thesis aims to contribute a portion of that scrutiny.

1.2 Anonymity in 5G Error Correction Coding

In order to protect the privacy of individuals using 5G networks, the standard uses a complex scheme to keep data transmissions anonymous. In the specifications, mobile devices utilizing the network are called User Equipment (UE). When a UE first connects to the network it uses a handshake procedure on public Broadcast Channels and Random Access Channels that allow the network to assign to a UE an identifier that the network can address the UE by [1]. This identifier is ostensibly random and temporary to help keep eavesdroppers from being able to associate data traffic with a particular UE. This identifier is called a Radio Network Temporary Identifier (RNTI). There are several varieties of RNTI, however, all of them help to protect the mobile subscriber UE from eavesdroppers. One of the ways these identifiers serve their purpose is in the error-correction coding utilized by standard.

Two methods are used to help correct erroneous transmissions: Polar coding and Low-Density Parity Check (LDPC) codes [4]. Both Polar coding and LDPC use the RNTI to prevent unintended recipients from being able to read the traffic transmitted. The error correction coding schemes use the unique RNTI as an input to a function used to "scramble" the rest of the coded message. This scrambling means that UEs without the RNTI used will be unable to decode the message, as it will have too many errors (produced by the scrambling). In this way, eavesdroppers of 5G traffic will be unable to decipher any part of a message not addressed to them, yielding privacy gains for the intended recipients.

Recent research has shown this combined system of addressing, error-correction encoding, and enciphering to be flawed, however [5]. That research examines the Polar coding implementation in the specifications for the Physical Downlink Control Channel (PDCCH) and found methods to associate a RNTI with the scrambled message [5]. Finding the RNTI for a transmission would allow an attacker to unscramble traffic at the physical and logical layers. This compromise of PDCCH is significant because the channel carries Downlink Control Information (DCI) for other downlink channels [1]. DCI contains channel management information such as scheduling slot format for other downlink channels. Unscrambling PDCCH helps an attacker to associate DCI with a particular RNTI and may allow them to identify other downlink channels with the same RNTI from that compromised DCI. This thesis will build on that work by investigating if a similar attack on the anonymity of data encoded with LDPC is possible, and evaluate which method leads to a safer and more secure 5G cellular network.

Just as Gardner used the PDCCH to examine the anonymity provided by the Polar coding, the Physical Downlink Shared Channel (PDSCH) specifications will be used to help determine the security of the RNTI in LDPC encoded data. The PDSCH is one of the physical layer channels of the 5G protocol used by the network to send to a UE user data and higher-layer control messages and is managed by PDCCH [1]. This makes it especially appealing for study, as it carries the banking transactions, emails, and other higher-layer data subscribers are often most directly concerned about protecting. Furthermore, the channel is designed to be flexible to a variety of transmission schemes and varying data rate requirements which mean the standards for PDSCH may be able to be used for other channels as well.

1.3 Thesis Overview

This thesis will first provide background concepts on the fundamentals of linear block codes, Polar coding, then LDPC codes and the PDSCH encoding process in Chapter 2. Next, three different attack models will be analyzed to determine if the PDSCH implementation is vulnerable to those attacks in Chapter 3. Those attacks include brute-force attack, known plaintext attack, and an analysis of multiple messages scrambled with the same scrambling sequence. Chapter 4 will discuss the performance of each of those attack vectors, and compare the strength of the anonymity provided by LDPC with that of Polar coding. Chapter 5 presents key points from the research, draws connections to similar scholarly work on LDPC coded systems as well as other applications of this research, and identifies areas for further research.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background

Examining the anonymity of the implementation of LDPC in 5G requires an understanding of LDPC encoding, decoding, error detection and correction. Effective comparison between the performance of attack methods against LDPC and Polar Coding in the 5G specification will also require some insight into the fundamentals of Polar coding. This chapter will introduce those concepts with a focus on relating them to the attack methods that will be discussed in Chapter 3. Finally, this Chapter will also describe how the encoding process for the PDSCH focusing on how the LDPC is implemented, and how the RNTI may be found from it.

2.1 Linear Block Codes

A linear block code is a coding scheme in which a block of k message digits is transformed into a block of n codeword digits, where each of the n codeword digits are from a given alphabet of elements, and are specified by a linear combination of the k message digits [6]. In binary communications, the alphabet of message digits is simply $0, 1$, and thus will be referred to in this work as bits. This alphabet represents a finite field of two elements, or a Galois Field of order 2 (abbreviated GF(2)). All computation for this thesis will be done in GF(2), which is modular arithmetic with the modulus 2. Furthermore, coding schemes are often referred to as an (n, k) code to describe their block size and composition. A code with seven-bit codewords and 4 message bits used to create each codeword could be called a (7,4) code.

2.1.1 Encoding

A linear block code is usually specified by a pair of corresponding non-invertible matrices called the generator matrix and parity-check matrix [6]. The generator matrix specifies the linear combination of message bits that produce the codeword bits. For a given vector of message bits $\mathbf{m} = [m_0, m_1, m_2, m_3..., m_k]$, and for a given generator matrix $\mathbf{G}$, the corresponding codeword $\mathbf{c}$ is produced by: $\mathbf{m}(\mathbf{G}) = \mathbf{c}$ [6]. To illustrate this and other coding

processes, consider a particular generator matrix $\mathbf{G}$ for a (7,4) Hamming code

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.1}$$

To encode the message $\mathbf{m} = [1011]$ it is multiplied be the generator matrix to produce a codeword

$$\mathbf{mG} = [1011] \cdot \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} = [0110011] = \mathbf{c}. \tag{2.2}$$

Using the generator matrix from the encoding example above, it's evident that the i-th bit in the codeword is the product of the message $\mathbf{m}$ and the i-th column in $\mathbf{G}$. The fourth bit of the codeword, for instance, can be written as:

$$c_3 = [m_0, m_1, m_2, m_3] \cdot \begin{bmatrix} G_{0,3} \\ G_{1,3} \\ G_{2,3} \\ G_{3,3} \end{bmatrix} = (m_0)G_{0,3} + (m_1)G_{1,3} + (m_2)G_{2,3} + (m_3)G_{3,3} = m_0 + m_1 + m_2 = 0 \tag{2.3}$$

which reduces to the sum of the first three message digits.

Conversely, the parity-check matrix $\mathbf{H}$ defines a complementary set of equations each bit of the codeword should satisfy if generated properly. The parity-check matrix shown below

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.4}$$

corresponds to the generator matrix used above.

To check the validity of a received codeword, a syndrome is computed by $\mathbf{H}(\mathbf{c^T}) = \mathbf{s}$. Since the parity-check matrix is made to evaluate the same equations the codeword was generated

with, a valid codeword should produce a syndrome of $\mathbf{0}$. Following the encoding example above, the syndrome is produced with matrix multiplication shown in Equation 2.5:

$$\mathbf{H}(\mathbf{c^T}) = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = [0,0,0]^T = \mathbf{s} \tag{2.5}$$

2.1.2 Error Correction & Decoding

In digital communications errors in the data can be induced by the medium during transmission. Thermal noise, multipath interference, and interference by other signals can all cause these errors. If an error is induced, some of the bits of the received codeword are modified in transmission. If the codeword no longer satisfies the imposed parity constraints of the code, the result is a non-zero syndrome which signifies the presence of the error. If the error modifies the codeword but still results in a null-vector syndrome, the error is considered undetectable [6]. Note that calculating the syndrome can detect errors, but does not inherently correct them. To correct any errors, the number and location of those errors must be obtained. Limited error correction is possible with the syndrome, and is generally the simplest form of error correction [6]. The received codeword (which may contain errors) may be considered the sum of the original codeword and an error vector $\mathbf{e} = [e_0, e_1, ...e_n]$. Calculating the syndrome $\mathbf{s}$ can be shown to be a function of the error vector only as in Equation 2.6

$$\mathbf{H}(\mathbf{c}+\mathbf{e}) = \mathbf{Hc} + \mathbf{He} = \mathbf{0} + \mathbf{He} = \mathbf{s}. \tag{2.6}$$

This holds for any codeword $\mathbf{c}$ since for any codeword $\mathbf{c}$, $\mathbf{H}(\mathbf{c}) = \mathbf{0}$. A syndrome for an (n, k) code will be $n-k$ bits, however, the code word is n bits. Therefore, for a binary system there are then 2^n possible error patterns, but only 2^{n-k} possible syndrome patterns. It is evident then, that there is not a one-to-one mapping of error patterns to syndromes. Below are two

different bit sequences (both derived from the same codeword shown above with different error patterns added) that are shown to have the same syndrome

$$\mathbf{H}(\mathbf{c^T}) = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = [1, 1, 0]^T = \mathbf{s} \tag{2.7}$$

$$\mathbf{H}(\mathbf{c^T}) = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = [1, 1, 0]^T = \mathbf{s}. \tag{2.8}$$

Specifically, there are 2^k error patterns with the same syndrome. Error correction based on the syndrome of a codeword is made possible by considering that for a channel where the probability of bit error (P_b) in the channel is less than 0.5, then for any given syndrome, the error pattern with fewest errors (or lowest hamming weight) is most probable. Because of this, syndromes may be listed in a table, and upon receipt of an erroneous codeword the syndrome with the lowest hamming weight that sums with the codeword to form a valid codeword (one that has a zero vector syndrome) is assumed to be correct. The codeword and error pattern are summed to correct the induced errors.

2.2 Polar Coding and Anonymity Vulnerability in 5GNR PDCCH

Polar codes are a type of capacity achieving linear block code [7]. Polar codes are defined by a generator matrix $G_N = G_2^{\otimes n}$ where N is the code length and $N = 2^n$, and $\otimes$ is the

Kronecker product. The example below is the polarization matrix for G_2

$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{2.9}$$

The value of this polarization matrix expansion to be used in the encoding process is that it allows bits to be separated into N virtual sub-channels where the reliability of each sub-channel exhibits known behavior. As the length of the encoding string increases, (as $N \to \infty$) those subchannels become completely unreliable (all noise, all message lost), or completely noiseless. Because these positions are known, data bits are placed in positions that will be mapped to noiseless channels, and noisy channels are filled with frozen bits (set to 0) [5] [7]. This also means there are known frozen bit positions with known values (0) to an eavesdropper, even without any a priori knowledge of the transmitted message. In [5], Gardner explores treating the scrambling sequences (which is applied after Polar coding similar to the PDSCH encoding process) as noise which corrupts the transmission. He goes on to outline a process for generating syndromes corresponding to those scrambling sequences. The syndromes are produced by the process below

$$\begin{aligned} s_0\mathbf{G_N} &= z_0 \\ s_1\mathbf{G_N} &= z_1 \\ s_2\mathbf{G_N} &= z_2 \\ &\dots \end{aligned} \tag{2.10}$$

in which each of the scrambling sequences (s) is multiplied with the polarization matrix of the corresponding size $\mathbf{G_N}$ where N is the code block length to obtain a syndrome (z).

By this process a table of syndromes that correspond to the scrambling sequences that were used to calculate them is created. Because of the well-defined characteristics of the Polar coding channels, when an eavesdropper receives a transmission that has been scrambled after polar coding, the attacker can identify the positions of all the frozen bits in the transmission. This allows the attacker to consider only the values of the frozen bits (which, if the codeword was not scrambled, would be zero), and to match them with bits in the syndrome table. By examining the syndrome values corresponding to the bit values in the frozen bit positions, the attacker can perform a search of the syndrome table to find a

syndrome with bit values that match the bit values in the frozen bit locations. However, the data bits cannot be guaranteed to match. This means false positives can arise. With a large selection of frozen bits, the success rate and probability of error can be managed by choosing a threshold for the size of the largest common substring of matching bits, or the Hamming distance between the received codeword and the syndrome. Gardner discusses the performance of his method in greater detail in Chapter 4 of [5]. This method allows a successive cancellation decoder and lookup table to identify the RNTI for a transmission given a low-noise channel.

2.3 Low-Density Parity-Check Codes

LDPC Codes are a specific class of linear block codes (the fundamentals of Linear Block Codes is described in 2.1) first proposed by Robert Gallager [8]. In LDPC Codes, the number of parity equations (each defining a bit in the codeword) and the number of message bits used in each of those equations is designed to be low in order to minimize computational complexity in the decoding process. In implementation, this leads to a parity-check matrix where most of the elements are 0. As an example, an LDPC parity-check matrix from the IEEE 802.11n standard is shown below in Figure 2.1 .

Figure 2.1. 802.11n compliant LDPC parity-check matrix

LDPC codes can be divided into two classes based on their parity-check matrix: they can be regular or irregular. For regular parity check matrices, each row has the same number of non-zero elements (always '1' for the binary case), and each column has the same weight as well [6]. In this way, each parity equation involves the same number of bits, and each bit appears in the same number of parity equations. Irregular LDPC codes have similarly sparse parity check matrices, however the weight of every row and column in the matrix is not uniform, and the parity equations are formed in a different fashion [6]. A few examples of irregular LDPC codes are those with parity check matrices formed by a pseudorandom process, or by permutations and concatenations of a base graph (protograph), or generated by a cyclic process.

Because these parity-check matrices can be large, and because viewing a matrix directly does not easily help to illuminate its important properties, they are often visualized by a Tanner graph. In a Tanner graph nodes in one row (circles below) represent bits of the codeword, and check nodes (squares below) in another row represent corresponding parity checks. Edges between nodes imply addition over GF(2). As a very simple example, consider a (3,1) code and its graph shown in Figure 2.2.

$$H = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

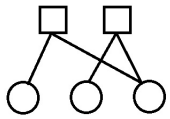

Figure 2.2. Example Tanner graph and parity-check matrix

The visualization helps to make apparent the parity checks that a valid codeword must be able to pass. In the example above, it's clear that the syndrome of a codeword for this code will be a vector of length two, the first element being the sum of the first and last bit in the codeword, and the second element being the sum of the second and last bit in the codeword.

In all of these cases, a primary benefit of LDPC codes is that the sparsity of the check

matrix translates to fewer bits involved in each parity check equation, and thus significantly fewer computations required to decode received codewords with errors present. The lower latency and simpler implementation of the codes while still providing relatively high levels of error correction capacity have made LDPC code-based schemes some of the closest to achieving Shannon capacity performances [9].

In what has been called "Gallager's Major Development", Gallager developed a novel decoding practice to accompany his codes [6]. He recognized that for any bit in the received codeword, solving a maximum likelihood equation (maximizing $P(v_n = b|\mathbf{y})$ where v_n is the nth bit in the transmitted codeword, and $\mathbf{y}$ is the received bit sequence, and $b \in 0, 1$) scales rapidly in complexity depending upon the sparsity (or lack thereof) of the parity-check matrix as well as the length of the codeword. In order to reduce this computational demand, he developed "message passing" algorithms known as the Sum-Product Algorithm and Belief Propagation Algorithm. These algorithms utilize an iterative feedback process that, for each bit, calculates the likelihood of satisfying a particular parity-check algorithm given the values of the other received bits in that check and "passes" it to the bit. These algorithms generally find the codeword that most closely matches the received bit sequence. Performance of these algorithms combined with soft decision demodulation increases rapidly as the block length increases [6]. Like most error correction coding schemes, though, the effectiveness rapidly diminishes for channels where the signal power compared to the noise at the receiver is very low. At decibel levels close to 0 the bit-error rate approaches 50% [6].

2.4 5GNR Physical Downlink Shared Channel Encoding Process

Data sent through the PDSCH encoding scheme is processed in multiple stages. At each stage additional processing is done and often additional bits are added to the higher-layer data. The sequence of these stages is shown below in Figure 2.3. This section will focus on the stages found to be most useful for the attacker- LDPC Encoding and Scrambling- and for simplicity will not consider the effects of rate matching and code block concatenation. In TS 38.212, 3GPP defines how the LDPC should be implemented for use in 5G. An examination of how the channel coding is performed will be followed by a discussion of how the scrambling is a function of the receiving UE RNTI [10]. This implementation is

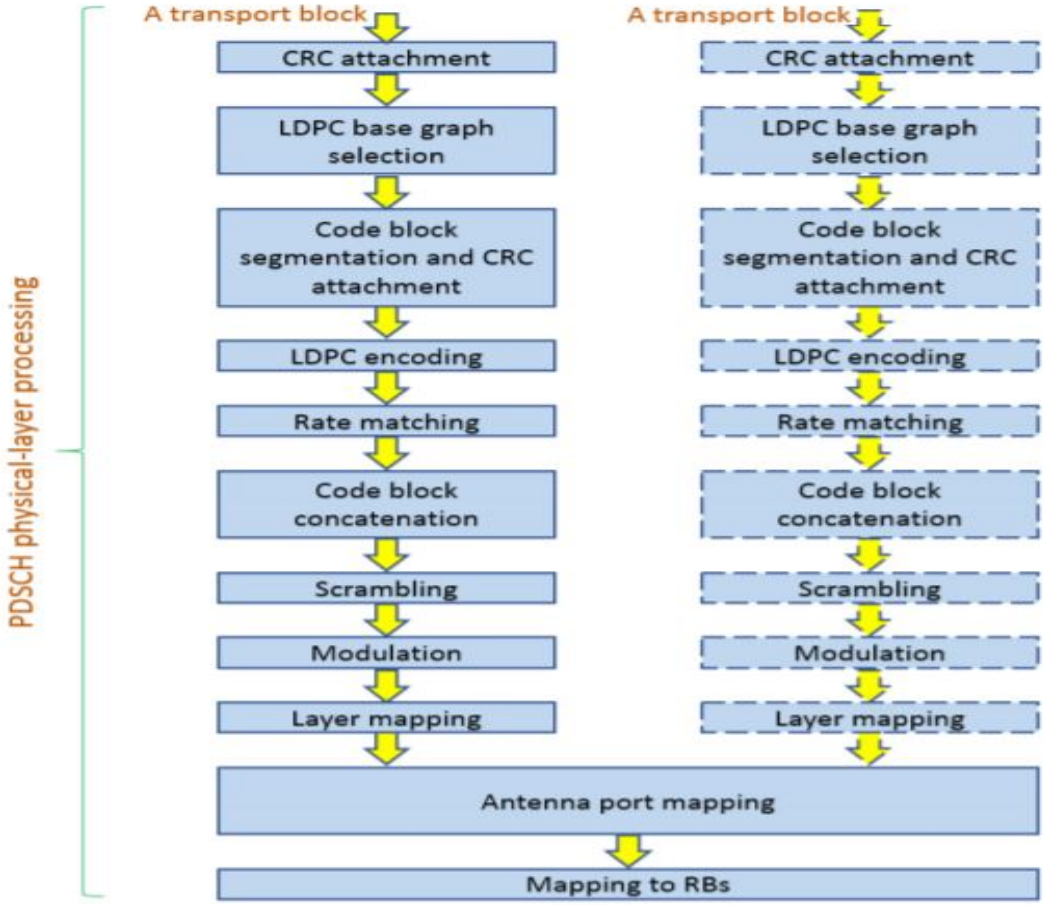

Figure 2.3. PDSCH Encoding Process Scheme
Source: Adapted from [1]

what future attack methods discussed in Chapter 3 are tested against. In PDSCH, a quasi-cyclic parity-check matrix is generated by expansion of a base graph [4]. This feature allows the channel to accommodate different data rates by scaling the block size of the code. Each code block is transmitted in a single resource block, meaning a larger code block size results in more data transmitted within the same period of time. There are two base graphs used, each of which is specified by a matrix of decimal numbers. The two graphs are similar, but the difference in the dimensions create two different code rates available from the base graphs. In order to obtain the parity-check matrix used for actual encoding, a "lifting size" (denoted by Z_c) is computed to ensure the parity-check matrix is expanded to properly fit the

input data block size. Then, each element in the base graph is replaced with a zero matrix of size Z_cxZ_c, or a Z_cxZ_c identity matrix, in which each column has been circularly shifted upward according to the integer in the base graph. A small base graph B is listed beside a corresponding expanded parity-check matrix using a lifting size of $Z_c = 2$ to illustrate the expansion sequence

$$B = \begin{bmatrix} 1 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad H = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.11}$$

This process transforms an mxn base graph into a $(Z_c \cdot m)$x$(Z_c \cdot n)$ parity-check matrix. Base graph 1 is 46x68, and base graph 2 is 42x52. The set of lifting sizes (Z_c) the standard allows for ranges from 2 to 384. This allows for variations in block size for base graph 1 between 92x136 up to 17664x26112. After the base graphs are expanded, the product is used as the parity-check matrix for encoding. The data to be transmitted is then encoded systematically with the parity-check matrix according to the equation

$$H \begin{bmatrix} c \\ w \end{bmatrix} = 0 \tag{2.12}$$

where c and w are the input data and set of added parity bits, respectively [4]. After encoding, the data is passed through a rate matching and interleaving scheme before being scrambled. Scrambling is accomplished using a pseudorandom scrambling sequence generated from a seed value and a length-31 Gold sequence. The sequence $c(n)$ is defined as shown in Equation 2.13

$$c(n) = (x_1(n + 1600) + x_2(n + 1600)) \text{mod } 2. \tag{2.13}$$

x_1 is initialized with $x_1(0) = 1, x_1(n) = 0$ for $n = 1, 2, ..., 30$ and defined according to:

$$x_1(n + 31) = (x_1(n + 3) + x_1(n)) \text{mod } 2. \tag{2.14}$$

x_2 is initialized with c_{init} such that $c_{init} = \sum_{i=0}^{30} x_2(i) \cdot 2^i$ and enumerated by:

$$x_2(n + 31) = (x_2(n + 3) + x_2(n + 2) + x_2(n + 1) + x_2(n)) \text{mod } 2. \tag{2.15}$$

c_{init} is defined by

$$c_{init} = n_{RNTI} \cdot 2^{15} + q \cdot 2^{14} + n_{ID} \tag{2.16}$$

in [10]. In this equation n_{RNTI} is the RNTI assigned to the mobile device, q is the codeword number ($q \in 0, 1$ to denote which of up to two codewords is being scrambled for the same transmission), and n_{ID} depends on higher-layer parameters. The seed value c_{init} is the only input to this pseudorandom sequence, and any number of digits from the sequence may be taken in order to match the code block length of the encoded bits. To illustrate this process, let us suppose a user with a $n_{RNTI} = 100$, a single code block ($q = 0$), and a $n_{ID} = 5$ is sent data over the PDSCH. We will assume the codeword for transmission is the same as the codeword calculated in the encoding example in Section 2.1.1 ($c = [0110011]$). In order to scramble this in the same fashion as the 5G standard dictates the sending unit would calculate the scrambling sequence $s = [c(0), c(1), ...c(6)]$ and would then add it to the codeword. In this case

$$c_{init} = 100 \cdot 2^{15} + 0 \cdot 2^{14} + 5 = 3,276,805 = (11001000000000000101)_2. \tag{2.17}$$

Using this value as the c_{init} to generate a 7-bit pseudorandom sequence yields $s = [0010101]$.

Once the scrambling sequence has been assembled, the encoded bit sequence and scrambling sequence are combined through bitwise addition modulo 2

$$\mathbf{y} = (\mathbf{s} + \mathbf{c}) \bmod 2 = [0100110]. \tag{2.18}$$

After scrambling the bits are modulated and transmitted.

This chapter has laid the foundations of syndrome decoding, LDPC, Polar coding, and the PDSCH process. Analysis showed syndrome decoding produces a syndrome dependent only on an error pattern, that LDPC decoding schemes work by maximizing the likelihood that a codeword was sent given the received bit string, and that Polar coding adds frozen bits with fixed values that make possible a table lookup sequence that can be used to identify the RNTI of a transmission. These conclusions are necessary for understanding the fusion of these concepts into a method attacking the anonymity of the C-RNTI used in 5G.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Attack Methodologies

3.1 Anonymity Attack Methodology

This chapter presents an analysis of anonymity attacks by an eavesdropper of LDPC-encoded data based on the model previously described for Polar coding in [5]. As a starting premise, only what is described in the 5GNR standard or can be received through an appropriately configured radio will be considered known to the attacker. The attacker's goal will be to identify the specific scrambling sequence used by the transmitter. This is because, as elaborated in Chapter 2, only the scrambling sequence is initialized with the RNTI as an input.

A study of the pseudorandom sequence generation algorithm was done to look for identifiable trends that may help to simplify the attacker's work. The algorithm as detailed in 2.4 utilizes only one initial value, which is translated into a 31-bit string from which the rest of the scrambling sequence is calculated [10]. Observing that the input space is therefore 2^{31} (or 2,147,483,648 possible values), 536,870,912 (one quarter of the total number of sequences) sequences were generated each 104 bits long (lowest codeword length based on expanded base graph 2). Figure 3.1 shows that each bit position was observed to take the value 1 or 0 with $p = 0.5$ (within $1.86 \cdot 10^{-9}$). This allowed each bit position in a scrambling sequence to be modeled as a Bernoulli random variable with $p = 0.5$ in later experiments. Similarly, the average weight (number of ones in the sequence) of the scrambling sequence set was shown to be half the length.

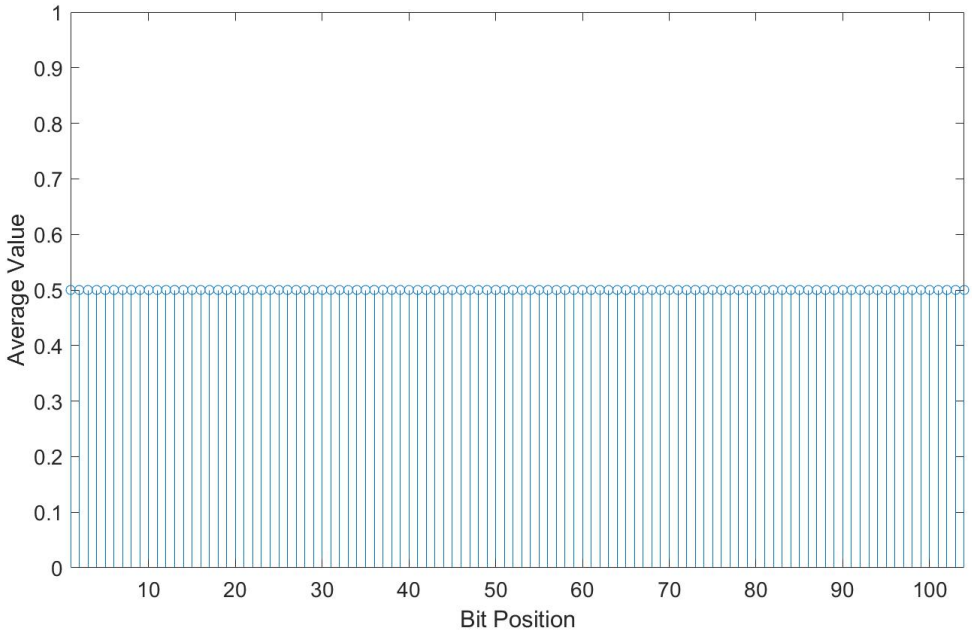

Figure 3.1. Average value for each bit position in 2^{29} scrambling sequences.

Standard LDPC decoding algorithms such as Weighted Bit Flipping or Sum-Product Algorithm will seek to find the codeword most likely used for the transmission. If an attacker were to treat the scrambling sequence as an error pattern, attempting standard decoding practices will unduly favor identifying scrambling sequences with lower weight. As identified from trials of 536,870,912 pseudorandom sequences, the actual sequences allowed in the 5GNR standard are designed such that on average the sequences consist of an even weighting of 1s and 0s. This means the premise of many decoding algorithms, that fewer errors is more likely, is not true in the scrambling sequence problem. Additionally, if each bit in the scrambling sequence is modeled as a Bernoulli random variable, then each bit of the codeword is flipped (receives an error) with $p = 0.5$. This can be modeled as a binary symmetric channel where the Signal to Noise ratio is 0 dB. Communications performance for LDPC coded systems in this environment is known to be poor [11]. In order to circumvent this problem, the attacker will have to build a method from the encoding and scrambling process alone.

The process of encoding and scrambling can be described mathematically as in Equation

3.1

$$\mathbf{y} = \mathbf{mG} + \mathbf{s}. \tag{3.1}$$

The equation represents the encoded and scrambled bit string intercepted by an eavesdropper as **y**, **s** is the scrambling sequence, and **m** is the original message, and **G** is the generator matrix that corresponds to **H**, which is the fully expanded parity-check matrix used for the transmission as dictated by the 5G standard. In this equation only **y** and the corresponding parity-check matrix **H** are considered known to the attacker. The scrambling sequence is of most consequence to an attacker concerned with associating the RNTI of a cellular device with the received traffic **y**.

In order to begin looking for an attack methodology, in the analysis below *s* will be treated as a simple bit string. Narrowing the scope of the variable to only 5G pseudorandom sequences as defined in [10] will be further discussed in 3.2. To eliminate an unknown (and unneeded) variable, the received bit string can be multiplied by the parity-check matrix that would be used to verify its satisfaction of parity-check constraints after un-scrambling. This technique treats the unknown bit string like any other error pattern (see 2.1.2) as thus applies Equation 2.6 to the PDSCH model in Equation 3.1. The multiplication also removes the encoded message from the equation as shown

$$\begin{aligned} \mathbf{Hy} &= \mathbf{H}(\mathbf{mG} + \mathbf{s}) \\ \mathbf{Hy} &= \mathbf{HmG} + \mathbf{Hs} \\ \mathbf{Hy} &= \mathbf{0} + \mathbf{Hs} = \mathbf{Hs}. \end{aligned} \tag{3.2}$$

These equations show that in the absence of other noise-induced errors, the product of an intercepted PDSCH transmission will be a syndrome entirely dependent on the bit string *s* (**Hs**). This process alone cannot sufficiently identify the bit string (or the RNTI as *s* is a valid pseudorandom scrambling sequence) used in the transmission. As shown in 2.1.2, multiple error patterns, or in this case multiple potential scrambling sequences, can produce the same syndrome. This can be gathered intuitively from equation set 3.2. For any bit string **s** that is added to the codeword **mG** and results in another codeword ((**m′**)**G**) then **Hy** will also be **0**. The fact that this process is non-injective can be formally proven from this intuition with the following argument.

Given $\mathbf{H}$ is an mxn parity-check matrix for a valid LDPC code, with a corresponding generator matrix $\mathbf{G}$ which is $(n-m)$xn, and any three 1x$(n-m)$ vectors $a \neq b \neq c$:

Proof the PDSCH syndrome problem is not injective

$$\text{let } s_1 = (bG - aG) \bmod 2$$

$$\text{let } s_2 = (cG - aG) \bmod 2$$

$$\mathbf{H}(a\mathbf{G})^T = \mathbf{H}(b\mathbf{G})^T = \mathbf{H}(c\mathbf{G})^T = \mathbf{0}$$

Then by adding the bit sequences and simplifying

$$\mathbf{H}(a\mathbf{G})^T = \mathbf{H}(b\mathbf{G} - (b\mathbf{G} - a\mathbf{G}))^T = \mathbf{H}(c\mathbf{G} - (c\mathbf{G} - a\mathbf{G}))^T.$$

$$\therefore \mathbf{H}(a\mathbf{G})^T = \mathbf{H}(a\mathbf{G} + s_1)^T = \mathbf{H}(a\mathbf{G} + s_2)^T. \tag{3.3}$$

This shows that there must exist multiple bit sequences that satisfy all of the parity checks for the code and will therefore be counted as false positives (excluding the solution actually used by the transmitter). This problem means that a one-to-one function that maps an encoded, scrambled message to a unique potential scrambling sequence (and therefore a unique RNTI) is not possible.

The factors and complications discussed above make conventional decoding methods ineffective at solving this problem consistently or accurately. In order to identify the RNTI used, three different methods utilizing the analysis above were examined for viability: brute-force computations, known plaintext attack, and cryptanalysis using multiple messages scrambled with the same sequence.

3.2 Evaluating a Brute-Force Attack

Brute-Force attacks work by starting with every possible solution to a problem, and then finding a method of verifying (or ruling out) each one before moving on to another. This is not usually efficient, however, it is often simple to implement, and can be used to gauge the scale and complexity of a problem. In 3.1 it was shown that any algorithm using the syndrome to solve the scrambling sequence problem must be able to account for multiple false positives. The lack of injectivity for the problem makes a brute-force solution appealing because attempting every possible solution ensures that every solution that satisfies parity-

check constraints (including the scrambling sequence actually used by the transmitter and the false positives) is found. The equation used to do this and an alternate form are derived from 3.2 and shown below

$$\begin{aligned} \mathbf{H}(\mathbf{y}+\mathbf{s}') &\stackrel{?}{=} \mathbf{0} \\ \mathbf{H}\mathbf{y} &\stackrel{?}{=} \mathbf{H}\mathbf{s}'. \end{aligned} \tag{3.4}$$

This equation will be evaluated for each and every possible solution(s'). Whenever the equality is found to be true, s' will be considered one of the brute-force solutions that may be a true or false positive.

To find characteristics that may be used by the attacker to eliminate the ambiguity in the brute-force solutions to Equation 3.2, additional analysis is necessary. Equation set 3.2 showed that in the absence of noise, the syndrome is dependent only upon the potential scrambling sequence. Nevertheless, in equation set 3.3 s_1 and s_2 (two different bit strings that both satisfy parity-check constraints) are defined in reference to two other codewords. It follows that a potential scrambling sequence can be defined for each codeword, and thus there are as many solutions as there are codewords. For an (n,k) LDPC code ($k = n-m$) there are 2^k codewords, and thus 2^k scrambling sequences that will satisfy all of the parity checks in the code. To investigate the problem graphically, trials were run to find the distribution of these brute-force solutions. In each trial a 20-bit message was generated as a sequence of bits chosen to be 0 or 1 with $p = 0.5$. This message was then encoded, and all of the brute-force solutions (and thus the true and false positive scrambling sequences in the context of PDSCH) that satisfied all of the parity checks were calculated using other valid codewords as described in equation set 3.3. Each of these brute-force solutions was then considered a binary number, and its decimal equivalent was added to an array from which a histogram was generated. The histogram results (shown in Figure 3.2) show an approximately uniform distribution of solutions to the problem.

Given a uniform distribution, if the brute-force solution is modeled as a random variable no useful conclusions can be drawn to limit the scope of the brute-force effort because all of the values within the range of possible scrambling sequence values is as likely to be the one used by the transmitter. In order to find all of the false positives (and thus ensure the true solution is also found) all of the potential scrambling sequences must be checked to see if

they, when added to the received codeword, satisfy all parity-check constraints.

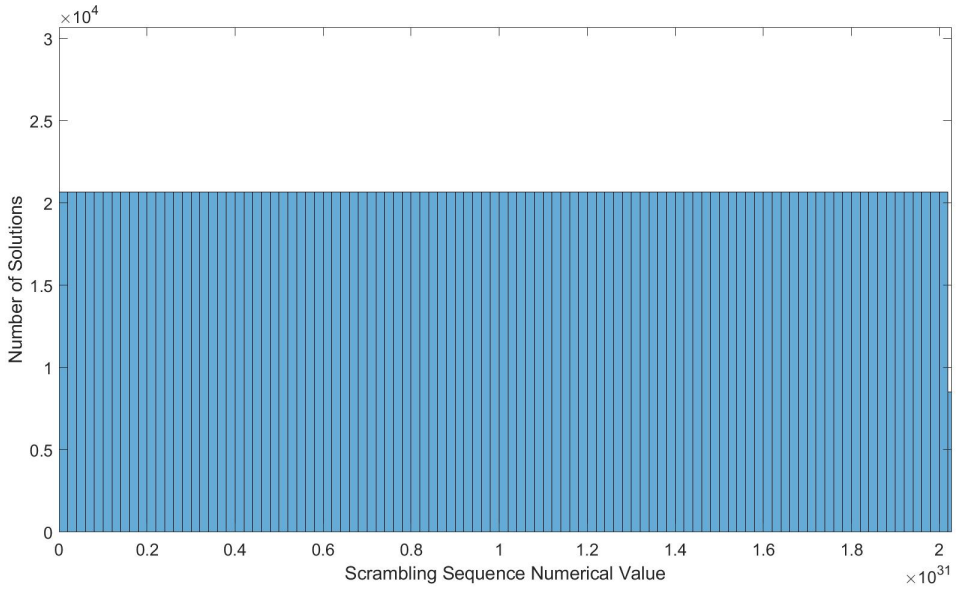

Figure 3.2. Histogram of brute-force solutions that satisfy parity-check constraints for randomly generated messages encoded by 5GNR LDPC base graph 2 with lifting size $Z_c = 2$

Considering that the attacker has no knowledge of the message, analysis above shows that the task of verifying which among all the brute-force solutions was actually used to encode the message is non-trivial. In order to examine the feasibility of this task, the specific implementation rules of the pseudorandom sequence from [10] were investigated. With a 31-bit seed value, a total of 2^{31} possible scrambling sequences are allowable by the standard. To examine if verifying the false positives identified through brute-force can be ruled out by comparison to the pseudorandom scrambling sequences allowed by the 5G standard, both types of scrambling sequences were modeled as uniform random variables. For a given bit sequence of length n, the sequence can take on 2^n possible values (0 to $2^n - 1$). The probability of any one of those pseudorandom sequences being one of the brute-forced solutions (one that satisfies 3.4) is presented as

$$\frac{\text{False Positives}}{\text{Total Bit Sequences}} = \frac{2^{n-m}}{2^n} = 2^{-m}. \tag{3.5}$$

Considering the 5GNR standard defined pseudorandom sequences to be a uniform random sampling from the same 2^n value set, the mean number of collisions between these values and the brute-force solutions can be expressed as the sum of the probability for each pseudorandom sequence that it will also be a brute-force solution. That means after checking all 2^{31} possible scrambling sequences allowed by the 5G standard, the mean number of values that would overlap with the brute-force calculated parity-check solutions would be $\frac{2^{31}}{2^m}$. For even the smallest lifting size on base graph 2, this would be $\frac{2^{31}}{2^{84}} = 1.11 \cdot 10^{-16}$. As the lifting size increases to accommodate longer block lengths of LDPC coded data, the dimension m of the parity-check matrix also increases, driving this expected number of collisions toward 0. This means that a brute-force solution is a valid approach to the problem only because of the constraints of the pseudorandom sequence per [10]. That is, calculating all 2^{31} possible pseudorandom scrambling sequences to check each one's satisfaction of the parity-check constraints is sufficiently unlikely to result in multiple false positives.

Another factor that can also be used by the attacker to reduce the number of required computations is Cyclic Redundancy Check (CRC) coding. As shown in Figure 2.3, before being encoded with LDPC, data is first fitted with a CRC checksum in order to help identify uncorrectable errors from the channel coding (LDPC). Analysis below will show how utilizing CRC encoding can reduce the number of brute-force solutions.

CRC encoding starts by representing the input bit string as a polynomial. Then the input bits are encoded using a generator polynomial which defines the properties of the particular CRC code [12]. Bits are then added to the message such that when the polynomial representing the message and appended bits are divided by the generator polynomial, the remainder will be $\mathbf{0}$ [12]. As an example, consider the generator polynomial $p = x^4 + x + 1$ and message $m = 1010110$. The message is then modeled as a polynomial where the degree of x is represented by the bit position and coefficient of x is represented by the bit value. Then polynomial long division is performed in GF(2) to find the CRC checksum, which is appended to the message. The example below shows the calculation of a checksum

$$\begin{aligned} p &= x^4 + x + 1 \\ m = 1010110 &= x^6 + x^4 + x^2 + x \\ \frac{x^6 + x^4 + x^2 + x}{x^4 + x + 1} &= x^2 + 1 + \frac{x^3 + 1}{x^4 + x + 1}. \end{aligned} \tag{3.6}$$

The checksum is the remainder $r = x^3 + 1$, which is then converted back into a bit sequence as before: $r = 1001$. The CRC encoded message is $[m, r] = 10101101001$.

CRC encoding represents added constraints on the message that, like the parity-check constraints, can be used to weed out false positives. Consider a CRC constructed codeword c, a scrambling sequence s, and a generator polynomial p

$$\frac{c+s}{p} = \frac{c}{p} + \frac{s}{p}. \tag{3.7}$$

Note that, c is a properly defined CRC-encoded message, and thus a multiple of p. Therefore, the remainder of $\frac{c+s}{p}$ is determined only by s. An algorithm can be devised to generate all of the possible scrambling sequences s that would contribute the CRC checksum derived from $\frac{s}{p}$.

For a given codeword of length n that is CRC encoded with a generator polynomial p of degree $l - 1$ (such that it will represent a bit sequence of length l).

1: A set of all multiples in GF(2^n) of the generator polynomial is calculated with degree $\leq n$ called **e** which is enumerated as:

$$\begin{aligned} e_0 &= 0 \\ e_1 &= 2^0 p \\ e_2 &= 2^1 p \\ e_3 &= 2^1 p + 2^0 p \\ e_4 &= 2^2 p \\ e_5 &= 2^2 p + 2^0 p \\ &\dots \\ e_{2^{n-l+1}-1} &= 2^{n-l} p + 2^{n-l-1} p + \dots + 2^0 p. \end{aligned} \tag{3.8}$$

2: The CRC checksum r is calculated from the scrambled codeword $c + s$ and added to each

element of **e** as shown

$$\begin{aligned} \frac{c+s}{p} &= a + \frac{r}{p} \\ e_0 &= e_0 + r \\ &\dots \\ e_{2^{n-l}} &= e_{2^{n-l}} + r. \end{aligned} \tag{3.9}$$

The error pattern set now contains every bit sequence that could cause the sum $c + s$ to have the checksum r, therefore $s \in \mathbf{e}$. As depicted in 3.8, **e** has 2^{n-l+1} elements, or $2^{n-l+1} - 1$ erroneous values to eliminate through brute-force comparison to 5G compliant scrambling sequences. This may seem inconsequential but for small values of n, a set of 2^{n-l-1} potential solutions in comparison to the total set of 2^{31} pseudorandom scrambling sequences may yield performance benefits over a brute-force algorithm that does not make use of CRC constraints.

3.3 Evaluating a Known Plaintext Attack

A known plaintext attack in cryptanalysis posits that an attacker knows the plaintext of the message being enciphered, and can use this information to attempt to unravel the cipher scheme for other messages. In the context of PDSCH, the LDPC encoding and scrambling obscure the original message in a way analogous to traditional cipher schemes. The known plaintext attack model was used to evaluate how knowing the error correction coded message would affect an attacker's ability to find the scrambling sequence (and thus identify the RNTI).

Until this point, all analysis has operated with the premise that the attacker has no a priori knowledge of the message being encoded. This has been fundamental to the challenge of the problem. As shown in Equation set 3.2, the received traffic y is made up of two variables unknown to the attacker- both the message and the scrambling sequence in the encoding. Section 3.2 evaluated a feasible attack method made possible by removing the unknown message from the equation using the parity-check matrix. Assuming the attacker has access directly to the message makes a brute-force approach unnecessary. Without the obstacle of the message, it can easily be shown the problem becomes trivial. The attacker receives y

by eavesdropping, then, with the knowledge of the message a, and the ability to encode the message a using the same parity-check constraints can reveal the scrambling sequence with simple addition as shown below

$$\mathbf{y} \coloneqq \mathbf{aG} + \mathbf{s}$$

$$\mathbf{y} + \mathbf{aG} = (\mathbf{aG} + \mathbf{s}) + \mathbf{aG} = \mathbf{s}. \tag{3.10}$$

Manipulating a UE into receiving a known data transmission may seem like a viable way to architect this circumstance. For instance, an attacker may be able to trick their target into downloading file contents known to the attacker. In reality, this is likely outside the ability of the attacker in the context of the whole 5G infrastructure. Higher-layer protocols in the 5G specification establish strong encryption of data travelling within the 5G network [13]. This feature alone means without the encryption keys even after coercing a UE to download a known file, the attacker would still not know what message was sent across PDSCH.

3.4 Evaluating the Use of Multiple Messages Encoded with the Same Scrambling Sequence

By eavesdropping an attacker may be able to obtain multiple messages that have been scrambled with the same scrambling sequence. In cryptanalysis, this often represents a vulnerability for the cipher scheme and may be advantageous to the attacker. This situation can be represented in the following equations:

$$\begin{aligned} \mathbf{aG} + \mathbf{s} &= \mathbf{y_a} \\ \mathbf{bG} + \mathbf{s} &= \mathbf{y_b} \end{aligned} \tag{3.11}$$

where $\mathbf{aG}$ and $\mathbf{bG}$ are two codewords such that $\mathbf{a} \neq \mathbf{b}$, and only $\mathbf{y_a}, \mathbf{y_b}, \mathbf{G}$ and the corresponding parity-check matrix $\mathbf{H}$ are known to the attacker. In this circumstance, no useful information is gained by the attacker. Linear combinations of the known values in GF(2) yield:

$$\begin{aligned} \mathbf{y_a} + \mathbf{y_b} &= (\mathbf{aG} + \mathbf{s}) + (\mathbf{bG} + \mathbf{s}) \\ \mathbf{y_a} + \mathbf{y_b} &= \mathbf{aG} + \mathbf{bG} + (\mathbf{s} + \mathbf{s}) = (\mathbf{a} + \mathbf{b})\mathbf{G}. \end{aligned} \tag{3.12}$$

This effectively eliminates the scrambling sequence (which is the attacker's goal in this case). Obtaining the syndrome from $\mathbf{y_a}$ and $\mathbf{y_b}$ both yield $\mathbf{Hs}$ as in the single message case. Similarly, any system of equations involving multiple messages in this fashion will see one more unknown variable (s) than the number of equations available to solve it with (one for each message sent).

The expression found above ($(a + b)G$) can be viewed as a third codeword for a message ($a + b$) that could be decoded by the attacker to obtain $a + b$, since no scrambling sequence is present. Finding a or b would be catastrophic to the security of the code as argued in 3.3. Analysis shows that using $a + b$ can provide a limited set of solutions to check for the possible values of a and b.

For a vector $c := a + b$ of length k, the number of possible bit sequences a, b that add to c and its bound can be expressed according to the equation shown

$$\sum_{i=0}^{k} \binom{k}{i} \leq (1 + k)^k. \tag{3.13}$$

With this method the attacker would find $c := a + b$, and generate every possible of the pair of values that satisfy the equation for c. These pairs would then be iterated through, with each potential solution for a denoted a' and the same convention for b and b'. In the event the attacker found the right addends, Equation 3.10 shows that encoding them and adding them to the received messages would yield the scrambling sequence as below

$$\begin{aligned} y_a + a'G &= (aG + s) + aG = s \\ y_b + b'G &= (bG + s) + bG = s. \end{aligned} \tag{3.14}$$

A reasonable test to evaluate if the current pair is the correct one might then be expressed as below

$$y_a + a'G \stackrel{?}{=} y_b + b'G. \tag{3.15}$$

Unfortunately for the would-be attacker, this test fails for all a' and b' pairs. Because this math is done in GF(2) the equations below show it can be manipulated to reveal it is a

tautology

$$\begin{aligned} y_a + a'G &\stackrel{?}{=} y_b + b'G \\ aG + s + a'G &\stackrel{?}{=} bG + s + b'G \\ aG + bG + a'G + b'G &\stackrel{?}{=} s + s \\ cG + cG &\stackrel{?}{=} s + s \\ 0 &= 0. \end{aligned} \tag{3.16}$$

Cipher systems that make use of the same key are generally less secure than ones that don't. The one-time pad is the iconic demonstration of this, where the scheme is provably secure unless the key is re-used. This is because for two messages m_1 and m_2 enciphered with the same key k allows an attacker to remove the key with simple addition

$$(m_1 + k) + (m_2 + k) = m_1 + m_2. \tag{3.17}$$

The sum $m_1 + m_2$ is considered much less secure than the individual ciphertexts, and is usually further attacked with frequency analysis or another cryptanalytic technique [14]. In this case the attacker is uninterested in the message compared to the key, and furthermore, Equation 3.13 shows the solution set for this sum is still larger than that of the pseudorandom scrambling sequences used in the brute-force approach. Lastly, the messages, in this case LDPC codewords, are approximately uniformly random leaving no avenue for pattern analysis. For this reason, while analyzing messages scrambled with the same scrambling sequence seems promising, the analysis above shows that receiving multiple messages sent using the same scrambling sequence does not simplify the problem of finding the transmitted scrambling sequence. This attack vector will therefore be disregarded in Chapter 4.

This chapter examined some of the challenges associated with developing a methodology attacking the anonymity of LDPC encoded data in PDSCH. Three different strategies for an attack were investigated: brute-force, known plaintext attack, and analysis of multiple messages scrambled with the same sequence. Brute-force was found to be the most robust in that it can be implemented consistently in all circumstances. Known plaintext attacks rely

on an assumption that the attacker knows the original message prior to encoding, and while the attack is extremely effective, the assumption was deemed unpractical at best. Analysis of multiple messages was found to yield no advantages compared to the brute-force case. These conclusions show that attacking the anonymity of PDSCH is possible, but is it practical? The next chapter will examine the performance of these attacks, and the requirements an attacker would have to go to execute them.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Performance

With an analysis of the efficacy of three different attack methodologies completed in Chapter 3, this chapter will quantify and compare the performance of the discussed attack models. First, each model will be evaluated regarding its potential for failure. Second, analysis will pursue optimization of any workable attacks before providing a comparison to the security of Polar coding as implemented in [10] and [4] and examined by [5].

4.1 Known Plaintext Attack Performance

A known plaintext attack on PDSCH represents a perfectly efficient and effective attack. Certain knowledge of the encoded message would make the problem of scrambling sequence identification (and thus RNTI identification) trivial as shown in Section 3.3. This presents the only attack vector examined that would feasibly allow the attacker to associate a RNTI to data channels in real time.

There is no potential for failure in the known plaintext attack as examined in Section 3.3. Furthermore, little need for optimization is present, since only a single LDPC encoding and vector addition operation are required. Fortunately for 5G users, the holy grail for the attacker- a priori knowledge of the encoded data or the ability to force specific data across the channel, represents a significant challenge, if possible. Avenues for guessing or choosing the encoded message were considered outside the scope of this research but represent opportunities for future vulnerability research.

4.2 Brute-Force Attack Performance

4.2.1 Brute-Force Attack Probability of Failure

Section 3.2 evaluated the efficacy of a brute-force attack on the LDPC encoded and scrambled communication scheme presented in the PDSCH. Section 3.2 also expressed the possibility that more than one of the 2^{31} 5G compliant scrambling sequences would satisfy all parity constraints to produce the same syndrome as the intercepted transmission, once

more adding ambiguity to the attacker's results. Precursory analysis showed this possibility seemed small. Analysis below will quantify the approximate probability of failure, reinforcing the conclusion that brute-force can associate a RNTI with a PDSCH transmission.

This analysis will model both the 5G pseudorandom sequences and the parity-check solutions to be uniformly distributed throughout the 2^n possible bit sequences for a code block of length n. By definition at least one pseudorandom sequence (the one used for the transmission) satisfies the parity-check sequence to produce the same syndrome as the intercepted transmission. Let p represent the probability of finding a parity-check solution, a string that produces the same syndrome as the actual scrambling sequence used. Removing the one solution guaranteed to exist (the actual scrambling sequence used by the sender), $p = \frac{\text{\# remaining parity-check solutions}}{\text{\# remaining total bit sequences}} = \frac{2^{n-m}-1}{2^n-1}$. The probability of finding one or more parity-check solutions in the remaining $2^{31} - 1$ pseudorandom sequences is then calculated. This is done by first finding the probability that a pseudorandom scrambling sequence is not a parity-check solution $(1 - p)$. In order for the algorithm to find a unique solution this event must occur for all of the $2^{31} - 1$ sequences. Using those premises, the calculation follows below

$$\begin{aligned} Pr[\text{one or more additional collisions}] &= 1 - Pr[\text{no additional collisions}] \\ &= 1 - (1-p)^{2^{31}-1}. \\ &= 1 - (1 - \frac{2^{n-m}-1}{2^n-1})^{2^{31}-1}. \end{aligned} \tag{4.1}$$

An approximation can be applied on the basis that for all lifting sizes 2^{n-m} is much smaller than 2^n. This approximation is expressed below

$$\frac{2^{n-m}-1}{2^n-1} \approx \frac{2^{n-m}}{2^n} = \frac{1}{2^m}. \tag{4.2}$$

Substituting this approximation into Equation 4.1 yields

$$Pr[\text{one or more additional collisions}] \approx 1 - (1 - \frac{1}{2^m})^{2^{31}-1}. \tag{4.3}$$

This probability is shown to be extremely small, so small that the math cannot be effectively

computed with 8-byte variables using IEEE Standard 754 for Floating Point numbers without significant truncation [15]. In order to visualize the effect of increasing code block size, the probability of p alone, not the probability of failure itself, is shown in Figure 4.1. This shows an attacker can justifiably claim that they will uniquely identify the scrambling sequence used in the communication, and can stop calculating additional sequences upon the finding the first one that satisfies parity-check constraints.

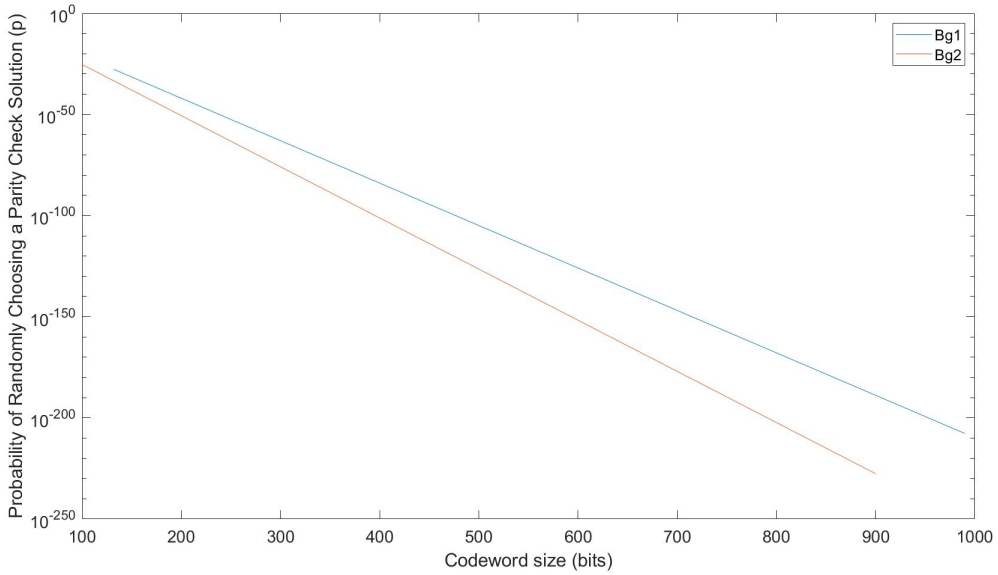

Figure 4.1. Probability of finding a scrambling sequence that satisfies parity-check constraints as block size increases.

4.2.2 Brute-Force Attack Time Complexity Optimization

This section will quantify the computational complexity of the brute-force algorithm. Time complexity is a type of computational complexity that expresses the nature of the required operations an algorithm will perform based on the size of the input (n) [16]. The premise of this technique is that elementary calculation steps (adding two numbers and storing the result, for example) requires an approximately fixed amount of time. An algorithm can then be evaluated by the asymptotic behavior of the amount of computations required for completion of the algorithm as the input size grows. This is commonly done using "big O notation", where an algorithm that grows linearly with the input n would be expressed as

$O(n)$, for example.

Applying this to the brute-force approach to de-anonymizing PDSCH illuminates two paths for minimizing the time complexity. The first is reducing the input size by reducing the number of potential syndromes to calculate, and the second is maximizing the efficiency of the algorithm (and thus minimizing the growth rate with respect to the input size).

Section 3.2 showed that for an (n, k) LDPC code there exist $2^k = 2^{n-m}$ different parity-check solutions, only one of which is the scrambling sequence used in transmission. The number of extraneous solutions therefore grows exponentially with respect to the lifting size of the parity-check matrix (which increases both n and m to accommodate a larger code block size) and is depicted in Figure 4.2.

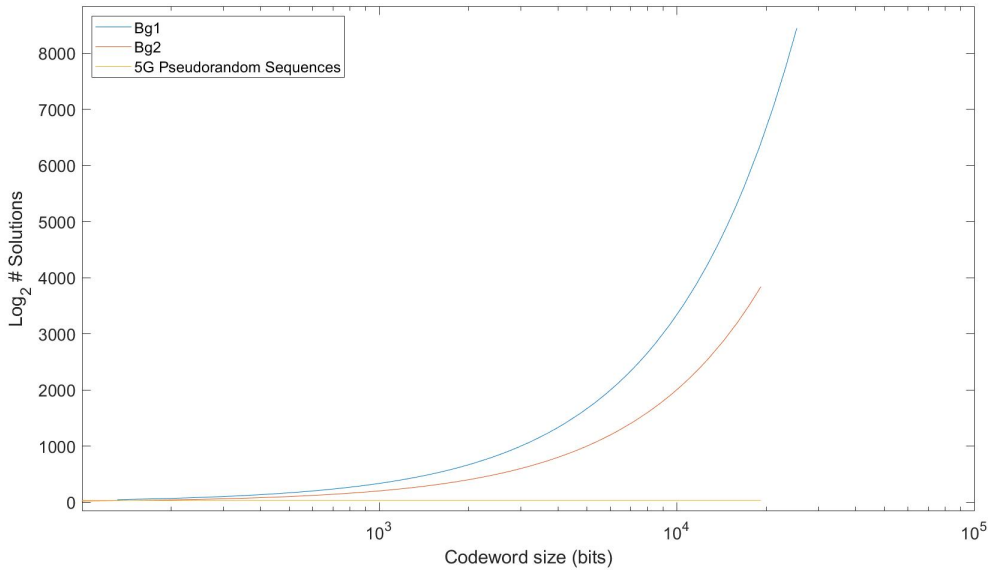

Figure 4.2. Exponential growth of parity-check solutions as code block size increases.

This growth characteristic makes a brute-force attack on an unspecified scrambling sequence of an LDPC encoded message incredibly costly. However, because the 5G standard only allows for 2^{31} possible scrambling sequences, the number of syndrome computations can be limited to this number. Figure 4.2 also shows graphically that the 2^{31} pseudorandom sequences is a fraction of the overall solutions, and does not scale based on the block size

used. This represents the smallest input size that can be used to minimize the time required to associate a PDSCH transmission with a RNTI using brute-force.

The basic time complexity of the algorithm is derived from how each individual pseudo-random scrambling sequence is generated and checked against the parity-check constraints. This operation is presented first in 3.4 and presented again here as Equation 4.2.2

$$\begin{aligned} \mathbf{H}(\mathbf{y} + \mathbf{s}') &\stackrel{?}{=} \mathbf{0} \\ \mathbf{H}\mathbf{y} &\stackrel{?}{=} \mathbf{H}\mathbf{s}'. \end{aligned} \tag{4.4}$$

The first form requires computing the sum of each scrambling sequence ($\mathbf{s}'$) with the received transmission($\mathbf{y}$) before the syndrome is computed (using the parity-check matrix $\mathbf{H}$) and compared to a zero vector. The second form, however, eliminates one step by comparing the syndrome of the scrambling sequence directly to the syndrome of the received transmission. Because the syndrome of the received transmission only needs to be computed one time, the second form is more efficient (no summation required).

With the chosen algorithm each step can be examined to determine the rate-limiting step and thus the overall time complexity. Equation 4.2.2 requires each scrambling sequence s' to be computed. Equations 2.13, 2.14, and 2.15 show that calculating each scrambling sequence requires a fixed number of arithmetic steps, and does not vary based on the number of scrambling sequences being computed. This means the step of calculating each scrambling sequence takes place with time complexity $O(n)$.

Next, the scrambling sequence must be multiplied by the parity-check matrix. Matrix multiplication algorithms for two non-square matrices having dimensions oxp and pxq have a time complexity equal to $O(opq)$ [17]. In the context of the brute force algorithm this makes the time complexity $O(pq)$, where the expanded parity-check matrix dimensions are pxq. This is an important point, as this means the time complexity of the matrix multiplication will grow as the code block size increases (which multiplies both dimensions by a factor of the lifting size Z_c) by the square of the lifting size. However, for a matrix of a constant size, the multiplication will require a fixed number of steps, and performing this multiplication for n different scrambling sequences will only result in a linear multiple of this number of steps. This means the time complexity of the matrix multiplication is only

linearly dependent on the number of scrambling sequences as a whole. For this reason, the time complexity of this step is shown as $O(nZ_c^2)$.

The last operation is bitwise comparison, which has a linear time complexity since bitwise comparison is an elementary operation.

The algorithm can be optimized slightly by changing the order of operations. As analyzed above, a scrambling sequence would be generated, then used to calculate a syndrome which is compared to the syndrome of the intercepted transmission $\mathbf{y}$. Instead, it is possible to check each bit of the scrambling sequence syndrome against that of the syndrome of $\mathbf{y}$ as it is calculated. The example below shows this process using the (7,4) Hamming code from Section 2.1.1, a scrambling sequence $\mathbf{s}'$, and an intercepted PDSCH transmission $\mathbf{y}$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{4.5}$$

$$\text{Let } \mathbf{Hy} = [1, 1, 1] \text{and let } \mathbf{s}' = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{4.6}$$

the first bit of the syndrome is calculated as the product of

$$\begin{bmatrix} H_{0,0} & H_{0,1} & H_{0,2} & H_{0,3} & H_{0,4} & H_{0,5} & H_{0,6} \end{bmatrix} \cdot \mathbf{s}' = [1010101] \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 1 \stackrel{?}{=} \mathbf{Hy}_{0,0}. \tag{4.7}$$

Since the equality holds, the same operation is done for the next digit of the scrambling sequence syndrome using the next row vector from **H**

$$\begin{bmatrix} H_{1,0} & H_{1,1} & H_{1,2} & H_{1,3} & H_{1,4} & H_{1,5} & H_{1,6} \end{bmatrix} \cdot \mathbf{s}' = [0110011] \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 1 \stackrel{?}{=} (\mathbf{Hy})_{0,1}. \quad (4.8)$$

Once more the equality is true, so the algorithm continues to the final bit

$$\begin{bmatrix} H_{2,0} & H_{2,1} & H_{2,2} & H_{2,3} & H_{2,4} & H_{2,5} & H_{2,6} \end{bmatrix} \cdot \mathbf{s}' = [0001111] \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 0 \stackrel{?}{=} (\mathbf{Hy})_{0,2}. \quad (4.9)$$

The last bit is found to be in error, and so the algorithm will iterate to the next scrambling sequence and try again. With this process, calculations can be stopped at the first parity-check constraint not satisfied by the scrambling sequence $\mathbf{s}'$. It can be estimated that this practice will, on average, allow each computation to be completed in half the time of the full matrix multiplication, with some syndromes failing on the first bit comparison, and some on the last. This does not change the asymptotic relationship of the required operations and the input size, however, the overall number of operations is reduced. Thus, while the time complexity is unaltered, the algorithm is nevertheless faster than the one initially described.

The rate limiting step responsible for the overall time complexity, is the matrix multiplication considered to be $O(nZ_c^2)$, showing that the required operations grows linearly with more scrambling sequences to calculate, but grows exponentially if done with larger code block sizes. With such a large input size ($2^{31} = 2,147,483,648$), even a linear time complexity is significant. However, especially as the lifting size grows, brute-force computations will be

found to be unmanageable. This is not feasible for an attacker looking to associate a data channel with a target in real time. Even post-facto computation would require significant time for all but the smallest lifting sizes.

4.2.3 Brute Force Attack Memory Requirements

Brute-force efforts must also consider memory requirements in addition to time and computational resources. The smallest block length permissible in the 5G LDPC specifications is 104 bits. Memory to store 104 bits for each of the 2^{31} scrambling sequences would require $104 \cdot 2^{31}$bits $\approx$ 26GB, with the largest block length set requiring more than 6TB. Most household computers would not be able to store this in Random Access Memory (RAM), requiring slow fetch operations from Read-Only Memory (ROM). For this reason, calculation of a standalone table with RNTI values associated to their pseudorandom sequences for use by the brute-force algorithm was considered inefficient and not attempted.

4.2.4 CRC Exploitation Performance

Analysis in Section 3.2 showed that CRC encoding could be leveraged to find a narrower solution pool than offered by the parity-check constraints alone. In 3.8 it was shown the number of solutions that satisfy the CRC constraints is dependent on the number of bits in the CRC encoded message k, as well as the degree of the CRC generator polynomial, denoted l. In the 5G LDPC standard k can be represented as the difference in the dimensions of the parity-check matrix. For an mxn matrix this means the total solutions to iterate through would be $2^{n-m-l+1} - 1$. In the PDSCH specification a generator polynomial of degree 24 is used, simplifying this expression to $2^{n-m-23} - 1$. This expression is notably smaller than the total solutions to the parity-check constraints (2^{n-m}). As the lifting size of the base graph increases, however, this number of solutions still grows larger than 2^{31}. This dynamic means that using the CRC constraints does not outperform brute-force computations using all of the pseudorandom sequences for 50 of 51 code block sizes using base graph 1, and 47 of 51 code block sizes using base graph 2. It can be concluded then, that iterating through each of the 2^{31} 5G compliant scrambling sequences remains the most robust attack model.

4.3 Comparison to Anonymity Vulnerability in Polar Coding

Performance analysis in previous sections have now facilitated the comparison of the examination of PDSCH for anonymity vulnerabilities to those found in PDCCH in [5]. In Section 3.3.1 of [5], Gardner states the total number of syndromes to be stored and searched through is 2^{15} compared to 2^{31} as implemented in PDSCH. This follows from the 5G specification for Polar coding, where a scrambling sequence is generated in a similar fashion to the LDPC case [10]. This reduction in required brute-force computations by a factor of 2^{16} and the added benefit that the Polar coding schemes are only utilized with code block sizes less than or equal to 512 bits is key to exploiting the anonymity vulnerability of PDCCH. The process Gardner presents to generate a syndrome table is fundamentally similar to the one utilized here- it also requires matrix multiplication, and bitwise addition. The result is a time complexity also related linearly to the number of scrambling sequences, and polynomially to the size of the polarization matrix. Analysis in this thesis showed finding the RNTI associated with a PDSCH transmission comes at a computational cost that outweighs the usefulness of the attack. However, in code blocks of a similar size as those utilized in PDCCH, and with a restricted scrambling sequence set, LDPC encoded data could be just as vulnerable. Conversely, at the scale employed by PDSCH the Polar coding scheme utilized by PDCCH would be similarly infeasible for an attacker to exploit in real time.

The performance and implementation of a brute-force attack on the anonymity of PDSCH was examined in this chapter. Analysis in Section 4.2.1 showed the probability of failure for the algorithm to be extremely low, however, time complexity analysis in Section 4.2.2 shows the time required to execute the algorithm to be prohibitively long for a useful attack by an eavesdropper. The same analysis showed the greatest contributing factor to the large processing time is the size of the LDPC parity-check matrix, which grows dramatically with regard to the lifting size. With this in mind, comparison to the vulnerability in the Polar coding of PDCCH in [5] found that the 5G standard constraints for PDCCH that limited polarization matrix size and number of pseudorandom sequences made the exploitation of a brute-force approach possible to execute.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusion

5G networks promise to make cellular data transfer possible at scales and speeds never before available to the mass populace [2]. Implementation for this technology has begun already, reaching nearly every part of the globe, and users of this technology are expected to increase by three billion by 2026 [2]. 3GPP has designed the specifications of this new network infrastructure with security in mind, however, ongoing research continues to verify and scrutinize the system to help keep the data traveling this network safe.

In 5G networks, user data is transported over a channel called PDSCH along with control messaging for higher-layer protocols [1]. PDSCH channel transmissions (like those of other channels) are emitted using radio waves in public settings, making intercept possible for a would-be eavesdropper. To combat this threat the 5G standard employs a method of scrambling transmissions with a sequence based on a unique identifier (called a RNTI) assigned to the intended recipient. However, analysis in [5] found that this process could be exploited in the 5G PDCCH channel, and this thesis examined the PDSCH channel for the same vulnerability.

A study of Linear Block codes, then additional study specific to LDPC was presented in order to effectively investigate the channel coding scheme of PDSCH specified in [4]. This investigation found the anonymity of a RNTI was protected in PDSCH transmissions encoded with LDPC as implemented by 3GPP 5G standard.

Observation of 2^{29} of these scrambling sequences showed the scrambling sequences could be effectively modeled by uniform random variables. Doing so allowed analysis in Chapter 3 to show that a brute-force attack that searched through every scrambling sequence would provide a unique solution to an attacker looking to de-scramble and associate a RNTI with the PDSCH transmission. Time complexity analysis presented in Chapter 4 showed that the time required to execute the brute-force operation outweighed its usefulness as an attack. The analysis also showed that the major contributors to this slow computation are the scale of the parity-check matrices used in the PDSCH standard, and the number of scrambling sequences an attacker must store or generate.

This research is consistent with similar work in the security field regarding large parity-check matrices such as those used in LDPC. The syndrome decoding problem of non-invertible LDPC matrices in various forms has been suggested as a replacement for problems involving prime factorization of large numbers in authentication and cryptographic schemes upon the development of a robust quantum computer [18]. Similarly, LDPC has been proposed for use in communications channels to afford a degree of encryption while providing error correction since at least 2000 [19].

This research has applications that spread beyond 5G PDSCH. Communications engineers already recognize LDPC has powerful error detection and correction capabilities [9]. This research demonstrates that wherever LDPC is implemented, additional security can be afforded to the transmission using a system similar to PDSCH, or otherwise dependent on the binary syndrome decoding problem to impose computational cost on would-be eavesdroppers. 3GPP could help to PDCCH secure from the vulnerabilities discussed in [5] by instead using LDPC encoding as implemented in PDSCH.

Opportunities to expand this research include the study of PDSCH from a high-layer perspective for ways of manipulating a UE into requesting or receiving a known data transmission, or the application of a scrambling sequence to physical layer protocols like 802.11n which already use LDPC for error correction.

List of References

[1] X. Lin, J. Li, R. Baldemair, T. Cheng, S. Parkvall, D. Larsson, H. Koorapaty, M. Frenne, S. Falahati, A. Grövlen, and K. Werner, "5G new radio: Unveiling the essentials of the next generation wireless access technology," 2018.

[2] Ericsson.com, "Ericsson mobility report," June 2021. [Online]. Available: https://www.ericsson.com/4a03c2/assets/local/mobility-report/documents/2021/june-2021-ericsson-mobility-report.pdf

[3] The European Parliament and the Council of the European Union. (2016, Apr. 27). Regulation (EU) 2016/679 of The European Parliament and of the Council. [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

[4] v. 3GPP TS 38.212, release 15, "Multiplexing and channel coding," June 2019.

[5] B. Gardner, "An efficient methodology to de-anonymize the 5G-new radio physical downlink control channel," M.S. thesis, Graduate School of Engineering and Applied Sciences, Monterey, CA, USA, 2020.

[6] F. Harris and B. Sklar, *Digital Communications: Fundamentals and Application*, 3rd ed. New York City, NY, USA: Pearson Education, Inc., 2021.

[7] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, July 2009.

[8] R. G. Gallager, "Low-density parity-check codes," M.I.T. Press, MA, USA, 1963. Available: http://www.inference.org.uk/mackay/gallager/papers/ldpc.pdf

[9] J. Hou, P. Siegel, and L. Milstein, "Performance analysis and code optimization of low density parity-check codes on rayleigh fading channels," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 5, pp. 924–934, 2001.

[10] v. 3GPP TS 38.211, release 15, "Physical channels and modulation," June 2019.

[11] A. N. F. Hamidi-Sepehr and G. Ermolaev, "Analysis of 5G LDPC codes rate-matching design," in *IEEE 87th Vehicular Technology Conference*, 2018. [Online]. Available: doi:10.1109/VTCSpring.2018.8417496

[12] W. W. Peterson and D. T. Brown, "Cyclic codes for error detection," *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961.

[13] Ericsson.com, "A guide to 5G network security," 2018. [Online]. Available: https: // www.ericsson.com/48fcab/assets/local/news/2018/10201291-04_gir_report_ broschure_dec2018_webb_181212.pdf

[14] M. N. S. A. Fort Meade, "The translations and KGB cryptographic systems," *The Venona Story*, pp. 26–27, 2009.

[15] "ISO/IEC/IEEE international standard - floating-point arithmetic," *ISO/IEC 60559:2020(E) IEEE Std 754-2019*, pp. 1–86, 2020.

[16] M. Sipser, *Introduction to the Theory of Computation*, 2nd ed. Boston, MA, USA: Thompson Course Technology, 2006.

[17] S. S. Skiena, *Sorting and Searching*. London: Springer London, 2008, pp. 103–144. Available: https://doi.org/10.1007/978-1-84800-070-4_4

[18] T. Tsuchida, M. Hirotomo, H. Ito, M. Takita, Y. Shiraishi, K. Nomura, M. Mohri, Y. Fukuta, and M. Morii, "A signature scheme based on the syndrome decoding problem using ldpc codes," in *2019 14th Asia Joint Conference on Information Security (AsiaJCIS)*, 2019, pp. 142–145.

[19] C. Monico, J. Rosenthal, and A. Shokrollahi, "Using low density parity check codes in the McEliece cryptosystem," in *2000 IEEE International Symposium on Information Theory (Cat. No.00CH37060)*, 2000, pp. 215.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California